

## Practical – 6

### Aim: Backtracking & Branch & Bound

6.1) You are given an integer N. For a given N x N chessboard. Implement a program to find a way to place 'N' queens such that no queen can attack any other queen on the chessboard.

A queen can be attacked when it lies in the same row, column, or the same diagonal as any of the other queens. You have to print one such configuration.

### Program Code:

```
import java.util.*;
import java.util.Queue;
import java.time.*;
import java.lang.*;
import java.io.*;
public class prac {
    private static boolean isSafe(int[][] board, int row, int col, int N) {
        int i, j;
        for (i = 0; i < col; i++)
            if (board[row][i] == 1)
                return false;
        for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
            if (board[i][j] == 1)
                return false;
        for (i = row, j = col; j >= 0 && i < N; i++, j--)
            if (board[i][j] == 1)
```

```
        return false;

        return true;
    }

    private static boolean solveNQueensUtil(int[][] board, int col, int N) {
        if (col >= N)
            return true;
        for (int i = 0; i < N; i++) {
            if (isSafe(board, i, col, N)) {
                board[i][col] = 1;

                if (solveNQueensUtil(board, col + 1, N))
                    return true;

                board[i][col] = 0;
            }
        }
        return false;
    }

    public static void solveNQueens(int N) {
        int[][] board = new int[N][N];

        if (!solveNQueensUtil(board, 0, N)) {
            System.out.println("Solution does not exist");
            return;
        }
    }
}
```

```
    }  
  
    printSolution(board, N);  
}  
  
private static void printSolution(int[][] board, int N) {  
    for (int i = 0; i < N; i++) {  
        for (int j = 0; j < N; j++)  
            System.out.print(board[i][j] + " ");  
        System.out.println();  
    }  
}  
  
public static void main(String[] args) {  
    int N = 4;  
    solveNQueens(N);  
}  
}
```

### Output:

```
PS D:\Probin's Work\Extra> javac prac.java  
PS D:\Probin's Work\Extra> java prac  
0 0 1 0  
1 0 0 0  
0 0 0 1  
0 1 0 0  
PS D:\Probin's Work\Extra> |
```

6.2) Amar takes 2, 6 and 7 hours of time to perform cooking,

gardening and cleaning respectively. Akbar takes 4, 8 and 3 hours of time to perform cooking, gardening and cleaning respectively. Anthony takes 9, 5 and 1 hours of time to perform cooking, gardening and cleaning respectively. Find out optimal job assignment for Amar, Akbar and Anthony.

**Program Code:**

```
import java.util.ArrayList;
import java.util.List;

public class prac {
    private static int[][] costs = {
        {2, 6, 7},
        {4, 8, 3},
        {9, 5, 1}
    };
    static int N = 3;
    static boolean[] assigned = new boolean[N];
    static int[] assignment = new int[N];
    static int[] minCostAssignment = new int[N];
    static int minCost = Integer.MAX_VALUE;

    private static void solve(int[] currentAssignment, int currentCost, int currentJob) {
        if (currentCost >= minCost)
            return;

        if (currentJob == N) {
```

```
minCost = currentCost;
System.arraycopy(currentAssignment, 0, minCostAssignment, 0, N);
return;
}

for (int i = 0; i < N; i++) {
    if (!assigned[i]) {
        assigned[i] = true;
        currentAssignment[currentJob] = i;
        solve(currentAssignment, currentCost + costs[i][currentJob], currentJob + 1);
        assigned[i] = false;
    }
}
}
```

```
public static void main(String[] args) {
    int[] currentAssignment = new int[N];
    solve(currentAssignment, 0, 0);

    System.out.println("Optimal Job Assignment:");
    for (int i = 0; i < N; i++) {
        System.out.println("Amar -> Job " + (minCostAssignment[i] + 1) + " (Cost: " +
            costs[minCostAssignment[i]][i] + " hours)");
    }

    System.out.println("Total Cost: " + minCost + " hours");
}
}
```



### Output:

```
PS D:\Probin's Work\Extra> javac prac.jav
PS D:\Probin's Work\Extra> java prac
Optimal Job Assignment:
Amar -> Job 1 (Cost: 2 hours)
Amar -> Job 3 (Cost: 5 hours)
Amar -> Job 2 (Cost: 3 hours)
Total Cost: 10 hours
PS D:\Probin's Work\Extra> |
```

**Conclusion:** From this practical I learned about the branch and bound approach.

**Staff Signature:**

**Grade:**

**Remarks by the Staff:**